# 1. Databázový systém

**Popište databázový systém- jeho části a úkoly, vlastnosti.**

Databázový systém (DBS) je systém pro správu a organizaci dat. Jeho účelem je usnadnit ukládání, přístup, aktualizaci a správu dat.

Měl by poskytnout:

* definici databáze,
* efektivní manipulaci databáze,
* ochranu dat,
* zotavení se z chyb systému.

Př. MatiaDB, MySQL, Oracle, PostgreSQL

Skládá se ze dvou částí:

1. ***Systém řízení báze dat*** - Program, který se stará o organizaci a o údržbu dat. (anglicky Database Management Systém )
2. ***Databázová aplikace*** – Program, který umožní manipulovat s daty uloženými prostřednictvím SŘBD.

**Systém řízení báze dat** nám umožňuje:

1. Definice dat (data definition) – poskytuje prostředky pro definování a uchování dat
2. Údržba dat (data maintenance) – udržuje data tak, že každému členu datové populace vyhrazuje záznam skládající se z položek, popisující dílčí informace o tomto členu.
3. Manipulace s daty (data manipulation) – poskytuje služby, které uživateli umožní vkládat, aktualizovat, rušit a třídit data v databázi.
4. Zobrazování dat (data display) – může poskytnout určité formy prezentace dat uživateli.
5. Integrita dat (data integrity) – poskytuje metody pro zajištění správnosti dat.

**b. Vysvětlete pojmy: systém řízení báze dat, báze dat, databázový systém, databázový model, integritní omezení, doména, redundantnost, primární klíč, cizí klíč, úrovně pohledu na data.**

**Systém řízení báze dat (DBMS):**

* Program, který se stará o organizaci a o údržbu dat.

**Báze dat:**

* Je organizovaný soubor dat, který je uložen v počítačovém systému a je snadno přístupný a aktualizovatelný.

**Databázový systém:**

* Je systém pro správu a organizaci dat. Jeho účelem je usnadnit ukládání, přístup, aktualizaci a správu dat.

**Databázový model:**

* Je abstraktní struktura, která popisuje, jak jsou data organizována a jak mohou být mezi sebou propojené. Příkladem je relační model, který využívá tabulky a vztahy mezi nimi.

**Integritní omezení:**

* Jsou pravidla a podmínky, které jsou definovány pro udržení konzistence dat v databázi. Může jít například o omezení hodnot v určitém sloupci nebo vazby mezi tabulkami.

**Doména:**

* Je množina možných hodnot, které může určitý atribut v databázi nabývat. Doména definuje rozsah platných hodnot pro konkrétní datový typ.

**Redundantnost:**

* Znamená opakování stejných dat v různých částech databáze. Minimální redundantnost je obvykle preferována, aby se zabránilo možným chybám a udržovala konzistence dat.

**Primární klíč:**

* Jedná se o jednoznačný identifikátor záznamu v databázi. Každý záznam v tabulce musí mít unikátní hodnotu primárního klíče.

**Cizí klíč:**

* Je klíč, který vytváří vazbu mezi dvěma tabulkami v databázi. Cizí klíč obvykle odkazuje na primární klíč v jiné tabulce a slouží k zachování integrity dat.

**Úrovně pohledu na data:**

* Označují různé úrovně abstrakce, které mohou uživatelé nebo aplikace vidět databázi. Úrovně pohledu umožňují skrýt složitost datové struktury a poskytují různé pohledy na data v závislosti na potřebách uživatele.

# 2. E-R model

**a. Popište E-R model- úlohu, značky pro kreslení, kardinalitu vztahů, integritní omezení.**

Úloha E-R modelu:

* E-R model slouží k vizualizaci a návrhu struktury databáze. Pomáhá identifikovat entity, vztahy mezi nimi a atributy, což umožňuje jasně definovat schéma databáze.

Značky pro kreslení E-R modelu:

1. **Entita**: Entita je objekt z reálného světa, o kterém uchováváme informace v databázi. (podstatné jméno)
2. **Atribut (\*,o)**: Atribut je vlastnost entity, která popisuje charakteristiky nebo vlastnosti entity. (např. jméno, příjmení, věk)
3. **Primární klíč (#)**: Unikátní pro každý záznam, zajišťuje, že žádný záznam nebude duplikován a bude jednoznačně identifikovatelný.
4. **Cizí klíč**: Cizí klíč je atribut, který navazuje vztahy mezi dvěma tabulkami v databázi.
5. **Vztah**: Vztah popisuje spojení mezi dvěma nebo více entitami, můžeme značit povinnost vztahu. (může, musí)
6. **Kardinalita vztahu**: Udává, kolik záznamů z entity 1 může být propojeno s kolika záznamy z entity 2 (1:N, M:N, 1:1)

**Integritní omezení:**

Jsou pravidla, která zajišťují konzistenci a platnost dat v databázi. Tyto omezení mohou být aplikována na jednotlivé položky (např. sloupce v tabulce) nebo na relace mezi různými tabulkami v databázi.

1. **Primární klíč (Primary Key)**: Unikátní pro každý záznam, zajišťuje, že žádný záznam nebude duplikován a bude jednoznačně identifikovatelný.
2. **Cizí klíč (Foreign Key)**: Definuje vztah mezi dvěma tabulkami. Hodnota v cizím klíči v jedné tabulce musí odpovídat hodnotě v primárním klíči v jiné tabulce. To zajišťuje integritu referenčních vztahů mezi tabulkami.
3. **Jedinečné omezení (Unique Constraint)**: Hodnota ve sloupci/sloupcích je unikátní v dané tabulce. To zabraňuje vzniku duplicitních záznamů v daném sloupci.
4. **Check Constraints**: Tyto omezení umožňují definovat pravidla, která musí být splněna při vkládání nebo aktualizaci dat v databázi. Např. věk zákazníka musí být větší 18
5. **Výchozí hodnota (Default Constraint)**: Toto omezení umožňuje nastavit výchozí hodnotu pro sloupec v případě, že není poskytnuta žádná hodnota při vkládání nového záznamu.
6. **Omezení nulové hodnoty (Not Null Constraint)**: Toto omezení zajišťuje, že určitý sloupec nemůže obsahovat hodnotu NULL, což znamená, že musí být vždy vyplněn.

# 3. Fáze definování databáze, normalizace

1. **Popište fáze definování báze dat.**

Analýza požadavků:

* + - Tato fáze zahrnuje sběr informací o potřebách uživatelů a oblastech podnikání, pro které bude databáze vytvořena. Identifikují se klíčové entity, vztahy mezi nimi a požadavky na data.

Návrh datového modelu:

* + - Na základě analýzy požadavků se v této fázi vytváří datový model, obvykle pomocí konceptuálních modelů, jako je Entitně-Relační (E-R) model. Definují se entity, vztahy a atributy, což pomáhá převést požadavky na konkrétní strukturu databáze.

Normalizace:

* + - Normalizace je proces, při kterém se databázový návrh optimalizuje tak, aby minimalizoval redundanci dat a zajistil konzistenci a integritu. Používají se různé normální formy, jako je první normální forma (1NF), druhá normální forma (2NF), atd.

Návrh fyzické struktury:

* + - V této fázi jsou definovány technické aspekty implementace databáze. Zahrnuje výběr databázového systému, volbu indexů, konfiguraci tabulek a dalších fyzických parametrů.

Dokumentace:

* + - Každá fáze definování databáze by měla být řádně zdokumentována. Dokumentace obsahuje popisy struktury databáze, vztahy mezi entitami, integritní omezení, a další důležité informace. Tato dokumentace slouží jako referenční materiál pro správu a vývoj databáze v budoucnosti.

Implementace + Testování:

* + - Poslední fází definování databáze je její implementace, kdy jsou vytvořeny tabulky, definovány klíče, integritní omezení a další prvky navržené v předchozích fázích. Tato fáze zahrnuje také naplnění databáze daty.

1. **Vysvětlete pojmy: entita, atribut, vztah.**

· **entita**

* + - * je objekt reálného světa, který je schopen nezávislé existence a je jednoznačně odlišitelný od ostatních objektů (tabulka)
      * entita je objekt reálného světa, který je předmětem našeho zájmu, o kterém má smysl uchovávat informace o entita je něco „čemu stojí za to dát jméno“ o například: student, kniha

· **instance entity –** konkrétní výskyt entity (záznam)

· **atribut –** rozumíme některou podstatnou vlastnost entity nebo vztahu (sloupec)

· **doména –** je množina homogenních dat atributu (datový typ)

· **identifikační klíč –** jeden nebo více atributů, které zajistí jednoznačnou rozlišitelnost entity (zajistí, že se v entitě nebudou vyskytovat dva stejné záznamy současně)

· **cizí klíč –** jedná se o atribut nebo skupinu atributů, které slouží k propojení entit

(v nadřazené entitě se jedná o primární klíč)

· **vztah –** je vazba mezi dvěma entitami (obecně i více entitami)

· záznam – jedná se o jeden řádek tabulky (neboli výskyt entity)

· primární klíč – označení identifikačního klíče v relačním modelu

· pole – jedná se o konkrétní výskyt atributu (tedy včetně pojmenování atributu), v tabulce je to sloupec i se záhlavím

· datový typ – jedná se o doménu z konceptuálního modelu, řeší i omezení pro domény (celá čísla, krátké datum, …)

1. **Popište převod konceptuálního modelu na fyzický model databáze.**

Převod konceptuálního modelu na fyzický model databáze je proces, který transformuje abstraktní strukturu a vztahy definované v konceptuálním modelu na konkrétní strukturu databáze s tabulkami, sloupci, klíči a omezeními. Zde je obecný postup tohoto procesu:

* 1. Identifikace entit a atributů: Začněte identifikací entit a atributů z konceptuálního modelu. Entitami jsou obvykle objekty, které mají v databázi reprezentaci, zatímco atributy popisují vlastnosti těchto entit.
  2. Mapování entit na tabulky: Každá entita z konceptuálního modelu je mapována na tabulku v fyzickém modelu. Každý řádek v tabulce pak představuje instanci této entity.
  3. Definice atributů jako sloupců: Atributy každé entity jsou převedeny na sloupce odpovídající tabulky. Každý sloupec má určitý datový typ a může mít různá omezení (například primární klíč, cizí klíč, hodnoty, které nepřijímá atd.).
  4. Identifikace klíčů: Pro každou tabulku je nutné identifikovat klíče, jako jsou primární klíče (PK), cizí klíče (FK) a další potřebné klíče pro zajištění integrity dat. Tyto klíče jsou následně definovány v tabulkách pomocí indexů a omezení.
  5. Určení vztahů: Vztahy mezi entitami jsou převedeny na vztahy mezi tabulkami pomocí klíčů. To zahrnuje definici cizích klíčů pro vazby mezi tabulkami.
  6. Normalizace: Zajištění normalizace databáze je klíčové pro efektivní a konzistentní ukládání dat. To zahrnuje rozdělení tabulek a sloupců tak, aby každý atribut byl uložen pouze jednou a zamezilo se tak redundanci dat.
  7. Stanovení integritních omezení: Dále je třeba definovat integritní omezení, jako jsou omezení referenční integrity (foreign key constraints), kontrolní omezení (check constraints) a další, které zajišťují konzistenci a platnost dat v databázi.
  8. Optimalizace: Nakonec může být provedena optimalizace fyzického modelu databáze, jako je vytvoření indexů pro rychlejší vyhledávání, rozdělení tabulek pro lepší správu paměti a další úpravy zaměřené na výkon a efektivitu databáze.

Celkově převod konceptuálního modelu na fyzický model databáze zahrnuje detailní technické rozhodnutí a implementaci, která transformuje abstraktní koncepty do konkrétního databázového schématu, které lze nasadit a používat pro ukládání a manipulaci s daty.

1. **Uveďte hlavní důvody pro normalizaci. Uveďte příklady**

Normalizace je proces organizování dat v databázi tak, aby byla minimalizována redundance a anomálie při aktualizaci, vložení nebo odstranění dat. Hlavní důvody pro normalizaci zahrnují:

**Minimalizace redundance**: Normalizace pomáhá minimalizovat redundanci dat tím, že je rozděluje do různých tabulek podle určitých pravidel, což snižuje opakování informací a udržuje databázi efektivní.

**Zachování datové integrity**: Normalizace pomáhá udržovat integritu dat tím, že minimalizuje šance na vznik nekonzistentních dat nebo anomálií, které mohou nastat při aktualizaci, vkládání nebo odstraňování dat.

**Snadnější údržba a správa**: Normalizovaná struktura databáze je obvykle snazší na správu a údržbu, protože každá tabulka obsahuje pouze související informace, což usnadňuje provádění změn a aktualizací.

**Lepší výkon**: I když normalizace může zvýšit nároky na výkon při provádění složitějších dotazů, často vede ke snížení množství dat, která jsou potřeba pro získání požadovaných informací, což může vést k rychlejším dotazům.

Příklady normalizace mohou zahrnovat:

* **První normální forma (1NF)**: Atribut, který může nabývat více hodnot najednou, musí mít vlastní entitu.
* **Druhá normální forma (2NF)**: Všechny ne-PK atributy musí být závislé na celém primárním klíči.
* **Třetí normální forma (3NF)**: Žádný ne-PK atribut nesmí být závislý na jiném ne-PK atributu.

**Řešení u všech**: Vytvoření nové entity.

# Databázové jazyky - jazyk SQL

*(Structured Query Language*) Je standardizovaný strukturovaný dotazovací jazyk, který je používán pro práci s daty v relačních databázích.

1. **Rozdělte databázové jazyky podle jejich činnosti, jazyk pro definici dat, jazyk pro manipulaci s daty, dotazovací jazyk.**

Databázové jazyky lze rozdělit podle jejich činnosti do tří hlavních kategorií:

* + Jazyk pro definici dat (**DDL** - Data Definition Language): Tento jazyk se používá k definici struktury databáze, jako jsou tabulky, sloupce, indexy atd.
  + Jazyk pro manipulaci s daty (**DML** - Data Manipulation Language): Tento jazyk se používá k manipulaci s daty v databázi, jako jsou vkládání, aktualizace, mazání atd.
  + Dotazovací jazyk (**DQL** - Data Query Language): Tento jazyk slouží k získávání dat z databáze pomocí dotazů.

1. **Popište syntaxi jazyků na příkladech.**

Jazyk pro definici dat (DDL - Data Definition Language) - Syntaxe na příkladu SQL:

|  |
| --- |
| CREATE TABLE Persons (  PersonID int,  LastName varchar(255),  FirstName varchar(255),  Address varchar(255),  City varchar(255) ); |

Jazyk pro manipulaci s daty (DML - Data Manipulation Language) - Syntaxe na příkladu SQL:

INSERT INTO Persons (PersonID, LastName, FirstName, Address, City)

VALUES (1, 'Doe', 'John', '123 Main St', 'Anytown');

Dotazovací jazyk (DQL - Data Query Language) - Syntaxe na příkladu SQL:

SELECT \* FROM Persons WHERE City = 'Anytown';

1. **Uveďte klíčová slova používaná v databázových systémech, datové typy jazyka SQL.**
   * + Klíčová slova:
     + CREATE - vytváří objekty v databázi (např. tabulky)
     + INSERT - vkládá nové řádky do tabulky
     + SELECT - vybírá data z tabulky
     + UPDATE - aktualizuje existující data v tabulce
     + DELETE - maže data z tabulky
     + WHERE - podmínka pro výběr řádků
     + FROM - určuje zdroj dat pro dotaz ● INTO - určuje cíl pro vkládání dat ● ...atd.
     + Datové typy v jazyce SQL:
     + INTEGER - celé číslo
     + VARCHAR(n) - řetězec o maximální délce n znaků
     + CHAR(n) - pevně dlouhý řetězec o délce n znaků
     + DATE - datum
     + TIME - čas
     + BOOLEAN - boolean hodnota (true/false) ● ...atd.

# 5. Jazyk SQL – příkaz SELECT

1. **Povinné a nepovinné části příkazu SELECT:**
   * + Povinné části:
     + Klíčové slovo SELECT: Určuje, že se jedná o dotaz na výběr dat.
     + Seznam sloupců nebo výrazů: Určuje, které sloupce nebo výrazy budou zobrazeny výsledku dotazu.
     + Klíčové slovo FROM: Určuje zdroj dat, tj. tabulky nebo výsledky poddotazů, ze kterých se budou data vybírat.
     + Nepovinné části:
     + Klíčové slovo WHERE: Určuje podmínky, které musí splňovat záznamy pro zařazení do výsledku dotazu.
     + Klíčové slovo GROUP BY: Určuje sloupce, podle kterých se mají data seskupit.
     + Klíčové slovo HAVING: Určuje podmínky, které musí splňovat seskupené záznamy pro zařazení do výsledku dotazu.
     + Klíčové slovo ORDER BY: Určuje řazení výsledku podle určených sloupců a směru.
     + Klíčové slovo LIMIT nebo FETCH: Určuje maximální počet řádků, které mají být vráceny.
2. **Vysvětlete výběr zobrazených sloupců- vytváření nových sloupců, přejmenování, výpočty, omezení pro řádky, omezení duplicit.**
   * **Vytváření nových sloupců:** Pomocí výrazů v SELECT klauzuli lze vytvořit nové sloupce, které mohou být výsledkem výpočtů, manipulace s textem nebo kombinace existujících sloupců.
   * **Přejmenování sloupců:** Pomocí klíčového slova AS lze přejmenovat sloupce výsledné tabulky.
   * **Výpočty:** Výrazy v SELECT klauzuli mohou obsahovat aritmetické operace, funkce a další operátory pro výpočty.
   * **Omezení pro řádky:** Klauzule WHERE umožňuje definovat podmínky pro výběr řádků.
   * **Omezení duplicit:** Pomocí klíčového slova DISTINCT lze odstranit duplicitní záznamy z výsledku dotazu.

SELECT DISTINCT column1, column2, ... FROM table\_name;

Vrátí pouze jedinečné záznamy.

1. **Uveďte povinné a nepovinné části příkazu SELECT týkající se řazení a seskupování dat.**
   * + Povinné části:
     + Klíčové slovo ORDER BY: Určuje, podle kterých sloupců se má výsledek řadit a v jakém pořadí.
     + Klíčové slovo GROUP BY: Určuje sloupce, podle kterých se mají data seskupit.
     + Nepovinné části:
     + Klíčové slovo HAVING: Určuje podmínky, které musí splňovat seskupené záznamy pro zařazení do výsledku dotazu.
     + Rozdíl mezi klauzulí HAVING a WHERE v jazyce SQL spočívá v tom, že WHERE se používá k filtrování řádků před jejich seskupením, zatímco HAVING se používá k filtrování seskupených dat.

SELECT age, AVG(score) AS avg\_score

FROM students

WHERE age >= 18

GROUP BY age

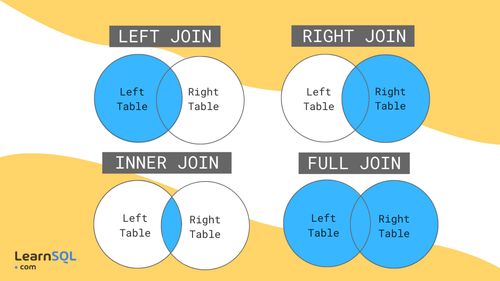
HAVING AVG(score) > 70;

V tomto případě klauzule WHERE filtrovala řádky, aby zahrnovaly pouze studenty ve věku 18 let a starší. Poté jsou data seskupena podle věku a klauzule HAVING filtruje seskupená data tak, aby zahrnovala pouze věkové kategorie s průměrným skóre vyšším než 70.

1. **Seskupování záznamů, vytváření skupin, agregační funkce:**
   * Seskupování záznamů: Seskupování umožňuje agregaci dat do skupin na základě hodnot v určitých sloupcích.
   * Vytváření skupin: Klíčové slovo GROUP BY se používá k vytvoření skupin pro seskupování záznamů.
   * Agregační funkce: Agregační funkce se používají k provádění výpočtů nad seskupenými daty v jedné nebo více skupinách. Mezi agregační funkce patří například SUM, AVG, COUNT, MIN, MAX.
2. **Popište typy spojení tabulek a jejich funkci.**

Existují různé typy spojení tabulek, včetně:

* + **Vnitřní spojení (INNER JOIN):** Vrátí pouze záznamy, které mají shodu v obou spojovaných tabulkách.
  + **Levé spojení (LEFT JOIN):** Vrátí všechny záznamy z levé tabulky a shodné záznamy z pravé tabulky.
  + **Pravé spojení (RIGHT JOIN):** Vrátí všechny záznamy z pravé tabulky a shodné záznamy z levé tabulky.
  + **Úplné spojení (FULL JOIN):** Vrátí všechny záznamy z obou spojovaných tabulek.



1. **Syntaxe příkazu SELECT a jeho použití:**

SELECT column1, column2, ...

FROM table1

WHERE condition

GROUP BY column1 HAVING condition

ORDER BY column1;

Tento příkaz vybere z tabulky table1 sloupce column1, column2, ... kde platí zadaná podmínka condition. Výsledek bude seskupen podle hodnot ve sloupci column1, a to pouze pro ty záznamy, které splňují podmínku pro seskupení (HAVING). Nakonec budou výsledky seřazeny podle sloupce column1.

1. **Použití vnořených dotazů vracejících právě jednu hodnotu:**

Vnořené dotazy, které vracejí právě jednu hodnotu, jsou často používány pro získání hodnoty, která se použije jako podmínka nebo součást výrazu v hlavním dotazu. Například:

SELECT column1, column2, (SELECT MAX(column3) FROM table2) AS max\_value FROM table1;

V tomto příkladu vnořený dotaz (SELECT MAX(column3) FROM table2) vrátí jedinou hodnotu - maximální hodnotu z column3 z tabulky table2. Tato hodnota je poté použita jako sloupec s názvem max\_value v hlavním dotazu.

**h. Použití vnořených dotazů vracejících více hodnot a operátorů ANY, SOME, ALL, IN, EXISTS:**

Vnořené dotazy, které vracejí více hodnot, jsou často používány s operátory jako jsou ANY, SOME, ALL, IN, EXISTS pro porovnání hodnot nebo s výrazy IN nebo EXISTS pro porovnání výsledků s jinými hodnotami nebo podmínkami.

* **Operátor ANY/SOME/ALL:**
* ANY nebo SOME: Porovnává hodnoty z výsledku vnořeného dotazu s hodnotami ve sloupci v hlavním dotazu. Vrací TRUE, pokud je alespoň jedna hodnota ze vnořeného dotazu splněna.
* ALL: Porovnává všechny hodnoty z výsledku vnořeného dotazu se všemi hodnotami ve sloupci v hlavním dotazu. Vrací TRUE, pokud jsou všechny hodnoty ze vnořeného dotazu splněny.
* **Operátor IN:**
* IN: Porovnává hodnoty ze sloupce hlavního dotazu s výsledkem vnořeného dotazu. Vrací TRUE, pokud je hodnota ze sloupce hlavního dotazu obsažena v seznamu hodnot vrácených vnořeným dotazem.
* **Operátor EXISTS:**
* EXISTS: Vrací TRUE, pokud vnořený dotaz vrátí alespoň jeden záznam. Používá se často s poddotazy, kde se ověřuje existence záznamu, a ne tak přesné porovnání hodnot.

# 6. Jazyk pro definici dat DDL

Se používá k vytváření a modifikaci struktury databáze, včetně tabulek, indexů, pohledů atd. Zde je popis tvorby databáze a tabulek s příslušnou syntaxí:

1. **Popište tvorbu databáze, tabulek, uveďte syntaxi příkazů.**

Vytvoření nové databáze se provádí pomocí příkazu **CREATE DATABASE:**

CREATE DATABASE databasename;

Vytvoření nové tabulky se provádí pomocí příkazu **CREATE TABLE.** Zde je příklad syntaxe:

CREATE TABLE tablename ( column1 datatype PRIMARY KEY, column2 datatype, column3 datatype,

...

);

Definice cizího klíče:

|  |
| --- |
| CREATE TABLE tablename ( **column1** datatype, column2 datatype,  FOREIGN KEY (**column1**) REFERENCES othertable(**column1**) ); |

1. **Popište změny tabulek a jejich odstranění, uveďte syntaxi příkazů.**

Přidání nového sloupce do existující tabulky se provádí pomocí příkazu ALTER TABLE:

ALTER TABLE tablename

ADD columnname datatype;

Změna datového typu existujícího sloupce se provádí také pomocí příkazu ALTER TABLE:

ALTER TABLE tablename

MODIFY COLUMN columnname new\_datatype;

Odstranění existujícího sloupce z tabulky se provádí pomocí příkazu ALTER TABLE:

ALTER TABLE tablename DROP COLUMN columnname;

Odstranění celé tabulky z databáze se provádí pomocí příkazu DROP TABLE:

DROP TABLE tablename;

Přidání cizího klíče k existující tabulce:

**ALTER TABLE tablename ADD CONSTRAINT fk\_constraint\_name FOREIGN KEY (column1) REFERENCES othertable(column1);**

Přidání primárního klíče:

ALTER TABLE tablename

ADD CONSTRAINT constraint\_name PRIMARY KEY (column1);

# 7. Jazyk pro manipulaci s daty DML

1. **Postup pro přidávání, odstraňování a změnu dat v databázi:**

Přidání nových záznamů do tabulky se provádí pomocí příkazu INSERT INTO:

INSERT INTO tablename (column1, column2, ...) VALUES (value1, value2, ...);

Odstranění záznamů z tabulky se provádí pomocí příkazu DELETE FROM:

DELETE FROM tablename WHERE condition;

Aktualizace existujících záznamů v tabulce se provádí pomocí příkazu UPDATE:

UPDATE tablename

SET column1 = value1, column2 = value2, ... WHERE condition;

1. **Charakterizace různých typů dotazů:**

Aktualizační dotaz (**UPDATE**):

* + Aktualizační dotaz slouží k aktualizaci existujících záznamů v tabulce. Pomocí něj můžeme měnit hodnoty ve vybraných sloupcích podle určené podmínky.

Přidávací dotaz (**INSERT INTO**):

* + Přidávací dotaz je speciální typ vytvářecího dotazu, který slouží k vkládání nových záznamů do tabulky.

Odstraňovací dotaz (**DELETE FROM**):

* + Odstraňovací dotaz slouží k odstraňování záznamů z tabulky na základě zadané podmínky. Používá se k odstranění záznamů, které již nejsou potřeba nebo jsou chybné.

# 9. Návrh E-R modelu v prostředí Workbench

**a. Popište návrh E-R model pro prostředí Workbench- jeho úlohu, značky pro kreslení, kardinalitu vztahů, integritní omezení.**

**Kardinalita vztahu**: Udává, kolik záznamů z entity 1 může být propojeno s kolika záznamy z entity 2 (1:N, M:N, 1:1)

**Integritní omezení**

Jsou pravidla, která zajišťují konzistenci a platnost dat v databázi. Tyto omezení mohou být aplikována na jednotlivé položky (např. sloupce v tabulce) nebo na relace mezi různými tabulkami v databázi.

1. **Primární klíč (Primary Key)**: Unikátní pro každý záznam, zajišťuje, že žádný záznam nebude duplikován a bude jednoznačně identifikovatelný.
2. **Cizí klíč (Foreign Key)**: Definuje vztah mezi dvěma tabulkami. Hodnota v cizím klíči v jedné tabulce musí odpovídat hodnotě v primárním klíči v jiné tabulce. To zajišťuje integritu referenčních vztahů mezi tabulkami.
3. **Jedinečné omezení (Unique Constraint)**: Hodnota ve sloupci/sloupcích je unikátní v dané tabulce. To zabraňuje vzniku duplicitních záznamů v daném sloupci.
4. **Check Constraints**: Tyto omezení umožňují definovat pravidla, která musí být splněna při vkládání nebo aktualizaci dat v databázi. Např. věk zákazníka musí být větší 18
5. **Výchozí hodnota (Default Constraint)**: Toto omezení umožňuje nastavit výchozí hodnotu pro sloupec v případě, že není poskytnuta žádná hodnota při vkládání nového záznamu.
6. **Omezení nulové hodnoty (Not Null Constraint)**: Toto omezení zajišťuje, že určitý sloupec nemůže obsahovat hodnotu NULL, což znamená, že musí být vždy vyplněn.

# 10. Zpracování dat v MySQL, import, export databáze, pohledy, oprávnění

c. Popište úlohu pohledů v MySql vytvoření nového pohledu a úpravu stávajícího.

**Úloha pohledů:**

1. Pohledy vytvářejí virtuální tabulky na základě výsledku dotazu.
2. Poskytují možnost definovat složité dotazy a používat je opakovaně.
3. Umožňují omezit přístup k určitým částem dat v databázi.
4. Poskytují abstrakci nad daty, což zjednodušuje jejich použití pro uživatele.

**Vytvoření nového pohledu:**

Pro vytvoření nového pohledu v MySQL použijete příkaz CREATE VIEW. Zde je příklad syntaxe pro vytvoření pohledu: Příklad:

CREATE VIEW muj\_pohled AS

SELECT jmeno, prijmeni

FROM uzivatele

WHERE role = 'admin';

**Úprava stávajícího pohledu:**

Pro úpravu stávajícího pohledu v MySQL můžete použít příkaz CREATE OR REPLACE VIEW, který nahradí existující pohled novým definovaným pohledem.

CREATE OR REPLACE VIEW muj\_pohled AS

SELECT jmeno, prijmeni, email

FROM uzivatele

WHERE role = 'admin';

d. Popište vytvoření nového uživatele, oprávnění pro uživatele.

**Vytvoření nového uživatele:**

Pro vytvoření nového uživatele v MySQL použijete příkaz CREATE USER. Zde je příklad syntaxe:

CREATE USER 'novy\_uzivatel'@'localhost' IDENTIFIED BY 'heslo';

**Přidání oprávnění pro uživatele:**

Pro přidání oprávnění pro uživatele použijete příkaz GRANT. Můžete určit, na jaké databáze, tabulky a operace má uživatel povolení.

GRANT SELECT, INSERT, UPDATE ON nazev\_databaze.tabulka TO

'uzivatel'@'localhost';

Pro udělení všech oprávnění na všech databázích a tabulkách použijte \*.\*:

GRANT ALL PRIVILEGES ON \*.\* TO 'uzivatel'@'localhost';

11. Tabulka- obecné a vyhledávací vlastnosti v návrhovém zobrazení, relace

a. Popište návrhové zobrazení tabulek, definování polí, nastavení vlastností pro různé datové typy.

1. **Návrhové zobrazení tabulek:**

* Návrh tabulek v MS Accessu je přehledným grafickým rozhraním, kde můžete snadno vytvářet a upravovat tabulky.
* Můžete definovat názvy tabulek, primární klíče a vztahy mezi tabulkami pomocí grafického rozhraní.

1. **Definování polí:**

* Každá tabulka má pole, která uchovávají data.
* Při definování polí v MS Accessu můžete vybírat z různých datových typů, včetně textu, čísel, dat/datumů, logických hodnot a dalších.
* Kromě toho můžete nastavovat další vlastnosti polí, jako jsou délka pole, formát, implicitní hodnoty a další.

1. **Nastavení vlastností pro různé datové typy:**
   * Například:
     + Pro číselné datové typy můžete nastavit velikost pole (např. INT(10)), což ovlivní zobrazení dat výsledných dotazů.
     + Pro textové datové typy můžete nastavit maximální délku pole (např. VARCHAR(255)).
     + Můžete také nastavit další vlastnosti, jako je povolení nebo zakázání NULL hodnot, implicitní hodnoty, klíče, indexy atd.

b. Vysvětlete pojmy: primární klíč, vstupní maska, titulky, velikost pole, formát, ověřovací

pravidlo aj. Ukažte na příkladech

1. **Primární klíč:**

* Primární klíč je jedinečný identifikátor každého záznamu v tabulce.
* Označen klíčem

1. **Vstupní maska:**

* Vstupní maska je vzorec, který určuje formát, do kterého uživatelé zadávají data do pole.
* Například, pro telefonní číslo může být vstupní maska nastavena jako "(###) ###-####".

1. **Titulky:**

* Pomáhají uživatelům lépe porozumět obsahu jednotlivých polí.
* Příklad: V tabulce "Kontakty" může být pole s názvem "Telefon" s titulkem "Telefonní číslo".

1. **Velikost pole:**

* Velikost pole určuje maximální délku dat, která mohou být uložena v daném poli.

1. **Formát:**

* Formát určuje, jak jsou data v poli zobrazena.
* Například, pro datum může být nastaven formát "DD/MM/YYYY",

1. **Ověřovací pravidlo:**

* Ověřovací pravidlo je pravidlo, které definuje, jaká data jsou povolena nebo zakázána v daném poli.
* Př. pro pole známka: “Between 1 And 5“

12. Výběrové dotazy, dotaz s parametrem a křížový dotaz v MS Access

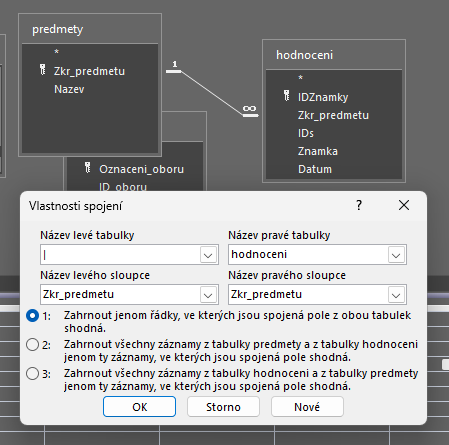
a. Popište návrhové zobrazení dotazu.  
Návrhové zobrazení dotazu v MS Accessu je grafické rozhraní pro vytváření dotazů. Umožňuje přidávání tabulek, výběr polí, definování podmínek, řazení a seskupování dat. Výsledky dotazu lze zobrazit v tabulkovém nebo listovém formátu a následně exportovat nebo uložit.

Obsah obrázku snímek obrazovky, Multimediální software, text, Grafický software

Popis byl vytvořen automaticky

b. Uveďte podmínky pro spojení tabulek v dotazu.

Idk asi stejný datový typ a musí být primární klíč v jedné z tabulek



Možná inner, right, left join

1. Popište agregační funkce, použití Tvůrce výrazů pro výpočty.

AVG,SUM,COUNT…

Obsah obrázku text, snímek obrazovky, číslo, Písmo

Popis byl vytvořen automaticky

1. Popište význam a použití výkonných dotazů.

* Výkonný dotaz je dotaz obsahující agregační funkci nebo propojení dvou a více tabulek.
* Slouží k rychlému získání dat s minimálním vlivem na výkon systému.

1. Vysvětlete rozdíly mezi výběrovými a výkonnými dotazy.
2. **Výběrové dotazy:**

* jsou zaměřeny na výběr konkrétních řádků nebo záznamů z tabulky podle určitých kritérií.



1. **Výkonné dotazy:**

* Zaměřují se na provádění výpočtů nebo agregačních funkcí nad daty.
* Mohou zahrnovat operace jako je spojování dat z různých tabulek, použití agregačních funkcí (např. **COUNT**, **SUM**, **AVG**, **MAX**, **MIN**), a řazení výsledků.



1. Charakterizujte aktualizační, vytvářecí, přidávací a odstraňovací dotaz.

Aktualizační dotaz (**UPDATE**):

* + Aktualizační dotaz slouží k aktualizaci existujících záznamů v tabulce. Pomocí něj můžeme měnit hodnoty ve vybraných sloupcích podle určené podmínky.

Vytvářecí dotaz (**CREATE**)

* + Slouží k vytváření struktury databáze.

Přidávací dotaz (**INSERT INTO**):

* + Přidávací dotaz je speciální typ vytvářecího dotazu, který slouží k vkládání nových záznamů do tabulky.

Odstraňovací dotaz (**DELETE FROM**):

* + Odstraňovací dotaz slouží k odstraňování záznamů z tabulky na základě zadané podmínky. Používá se k odstranění záznamů, které již nejsou potřeba nebo jsou chybné.

13. Formulář- vlastnosti v návrhovém zobrazení, vstupní formulář, makra

a. Popište vzhled, zobrazení, vlastnosti formuláře, typy formulářů (nekonečný, samostatný…),

soupravu nástrojů, návrhovou mřížku, seznam polí.

b. Popište tvorbu formuláře a vlastnosti jednotlivých sekcí.

c. Vysvětlete vkládání záznamů, textových polí, editaci dat, výpočtů

a nastavení vlastností.

d. Popište tvorbu vstupního formuláře. Vysvětlete nastavení vlastností.

14. Formulář s podformulářem, vstupní formulář a makra

a. Popište vzhled, zobrazení, vlastnosti formuláře s podformulářem, soupravu nástrojů,

návrhovou mřížku, seznam polí.

b. Popište tvorbu formuláře s podformulářem, propojení objektů. Vysvětlete podmínky pro

výpočty ve formulářích, vytváření nových polí.

c. Popište použití, vzhled, zobrazení, vlastnosti vstupního formuláře, soupravu nástrojů,

návrhovou mřížku. Vysvětlete vkládání záznamů, textových polí, editaci dat, výpočtů

a nastavení vlastností.

15. Sestava- vlastnosti - řazení záznamů, seskupování dat

a. Popište vzhled, zobrazení, vlastnosti sestavy, soupravu nástrojů, návrhovou mřížku, seznam

polí. Popište tvorbu sestavy a jednotlivých sekcí.

1. Vysvětlete tvorbu skupin, výpočty pro skupiny, průběžné součty.

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Popis byl vytvořen automaticky